

CMSC335

Web Application Development with JavaScript



Relational Databases/SQL

Department of Computer Science

University of MD, College Park

Slides material developed by Ilchul Yoon, Nelson Padua-Perez

Software Stacks

- Software stack - collection of software
 - Used to run dynamic Web Sites
 - Usually free/open-source
- LAMP Stack (Linux, Apache, MySQL, Perl, Python, PHP)
- MERN (MongoDB, Express, React, Node)
- MEAN (MongoDB, Express, Angular, Node)

Relational Database Systems

- Table (Relation)
 - Fundamental Unit of Storage
 - Rows/Columns(fields)
- Field Types
 - String
 - Integer
 - Float
 - enum
 - Etc.

Relational Database Systems

- Primary Keys
- Operations
 - Select
 - Project
 - Join
- Query Language Used
 - SQL (Structured Query Language)
- Goal of System Design - Avoid redundancy

MySQL/MariaDB Database System

- Database system included by XAMPP
- **MySQL/MariaDB Console**
 - Allow us to execute commands (e.g., queries, create tables, etc.)
- **Using MySQL provided by XAMPP**
 - Make sure you have started MySQL Server
 - » Use XAMPP Control Panel Application
 - We can issue SQL commands using the MySQL console
 - In a **PC** we start the console by executing `.\mysql.exe -u root -p`
 - » You can find mysql.exe at C:\xampp\mysql\bin
 - In a **Mac** you can start the console by executing `./mysql -u root -p`
 - » You can find mysql at /Applications/XAMPP/xamppfiles/bin
- **Notes:**
 - There is a test database called **test** in MySQL
 - We add **-u root** when starting the console otherwise we will have limited permissions
 - You do not need to specify any password (just press enter)

SQL (Structured Query Language)

- SQL allow us to create databases, tables, insert/delete data, etc.
- SQL allow us to perform the four basic operations of data storage:
 - Create, Read, Update, Delete (**CRUD**)
- SQL Intro
 - <https://www.cs.umd.edu/~nelson/classes/resources/web/resources/sqlReview.pdf>

SQL (Structured Query Language)

- **Commands end with semicolon and are non-case sensitive**
- **Showing databases available**
 - Example: **show databases;**
- **Creating databases**
 - Tables are part of databases, so we need to create a database
 - create database <NAME>;
 - **Example: create database ourDB;**
 - If everything went OK you will see something similar to:
Query OK, 1 row affected (0.02 sec)
- **Changing to a database**
 - Use <DATABASE_NAME>;
 - **Example: use ourDB;**

SQL Commands

- **Creating a table**
 - create table <tableName> (fieldList);
 - fieldList is a comma separated list of fields of the form
 - » <FIELDNAME> <TYPE> <ATTRIBUTES>
 - **Example: create table movies(name varchar(20), year int);**
- **Showing tables in a database**
 - **show tables;**
- **Looking at table structure**
 - describe <TABLENAME>;
 - **Example: describe movies;**

SQL Commands

- **Inserting data**

- insert into <TABLENAME> values (<COMMA_SEPARATED_VALUES>);
- **Example: insert into movies values ("Jaws Jr", 1976);**
- Strings in double or single quotes

- **Looking at table contents**

- select * from <TABLENAME>;
- **Example: select * from movies;**

Field Types (**Numeric/integer**)

- **Table Field Types**

- **int** - signed/unsigned integer.
 - » Aprox: (-2 billions \rightarrow 2 billions) or (0 \rightarrow 4) billions
- **tinyint** - signed/unsigned integer:
 - » (-128 \rightarrow 128) or (0 \rightarrow 255)
- **smallint** -signed/unsigned integer
 - » (-32768 \rightarrow 32767) or (0 \rightarrow 65535)
- **mediumint** - signed/unsigned integer:
 - » Aprox (-8 millions \rightarrow 8 millions) o (0 \rightarrow 16) millions
- **bigint** - signed/unsigned

Field Types (Numeric/floats)

- **Table Field Types**
 - **float(M,D)** - floating point
 - » M - Display length (total number of digits including decimals)
 - » D - number of decimals. M and D are not required and default to 10 and 2
 - » Decimal precision - 24 places
 - **double(M,D)** – floating point
 - » M - Display length (total number of digits including decimals)
 - » D - number of decimals. M and D are not required and default to 16 and 4
 - » Decimal precision - 53 places
 - » real is a synonym for double

Field Types (string/blob/enum/null)

- **Table Field Types**
 - **char(length)** - fixed-length string between 1 and 255 characters
 - » **Example:** state char(2)
 - **varchar(length)** - variable-length string between 1 and 255 characters
 - » **Example:** name varchar(20)
 - **blob or text** - (Binary Large Objects)
 - » Use to store binary data (e.g., images). Maximum size is 65535
 - **tinyblob or tinytext** - blob or text with maximum size of 255 characters
 - **mediumblob or mediumtext** - blob or text with maximum size of 16777215
 - **longblob or longtext** - blob or text with maximum size of 4294967295
 - **enum** - enumeration (maximum of 65535 values)
 - » **Example:** enum('M','F')
- **Strings can be specified with single or double quotes**
 - Use single quotes for strings if you want to be ANSI compatible
- **null is a possible field value**

SQL Commands

- Let's create another table
 - **create table friends (name varchar(20), met date, salary float);**
- Let's insert some values
 - **insert into friends values ("Mary", "2007-01-30", 10000);**
 - **insert into friends (name) values ("Jose");**

SQL Commands

- **Selecting data**
 - select * from <TABLE> where <CONDITION>;
 - » **Example:** `select * from friends where salary > 5000;`
 - **Comparison operators for conditions**
 - » = → equals
 - » != → not equals
 - » <= → less than or equal to
 - » < → less than
 - » >= greater than or equal to
 - » > greater than
 - **Logical Operators:** and, or
 - **Field specification (projection):**
 - select <FIELDLIST> from <TABLE> where <CONDITION>
 - » **Example:** `select name, met from friends where salary > 5000;`

SQL Commands

- **Deleting data from a table**
 - delete from <TABLENAME> where <CONDITION>;
 - DANGER - removes all the entries in the table
 - » delete from <TABLENAME>;
- **Removing a table**
 - drop table <TABLENAME>;
 - Example: **drop table movies;**
- **Removing a database**
 - drop database <DATABASENAME>;
 - Example: **drop database ourDB;**

SQL Commands

- **Field Attributes**

- primary key
- not null
- auto_increment

- **Let's create our table again**

create table friends (name varchar(20) primary key not null, salary float not null);

- **Autoincrement**

create table items (id int auto_increment primary key, name varchar(20));
insert into items (name) values ("house");

SQL Commands

- **Update**

- update friends set salary=20000 where name="Mary";
- Assuming there was a year field

» **Example:**

update friends set salary=7778, year=8 where name = "Pat";

- **Replace**

- If the record you are inserting has a primary key value that matches a record in the table the table record will be deleted and new one inserted
- **Example: replace into friends values ("Mary", 5000);**

SQL Commands

- **like** operator - to compare strings
 - % wildchar character - matches multiple characters
 - _ wildchar character - matches one character
 - **Example:** delete from friends where name like “%Jose%”;
- **Order by** - to display elements ordered by a field
 - **Example:** **select * from friends order by salary;**
 - » Output: elements will be listed in increasing salary order
 - **Example:** **select * from friends order by salary desc;**
 - » Output: elements will be listed in decreasing salary order
- **count** → allows you to determine number of records satisfying a criteria
 - **Example:** **select count(name) from friends where salary <= 12000;**
 - Output: number of friends satisfying salary restriction
- and, or, between operators

Access Commands

- **Accessing a remote database**
 - `mysql -h <HOST> -u <USER> -p`
 - Password must be provided after
- **Granting access via grant command**
 - `grant <PRIVILEGE_LIST> on <DATABASE>.* to <USER>@"%" identified by "<PASSWORD>";`
 - **Example:**
 - `grant all on myDB.friends to student@"%" identified by "goodbyeWorld";`
 - `<PRIVILEGE_LIST>` - all, create, delete, drop, insert, update

Additional Information

- The previous slides cover some fundamentals about SQL
- Slides that follow provide some additional information you can check at home

SQL Functions

- You can try the following functions by using **select** at the mysql console
- **Functions**
 - **lcase** - returns a lowercase string **Example:** `select lcase("TOM");`
 - **ucase** - returns an uppercase string **Example:** `select ucase("cat");`
 - **now()** - returns current date/time **Example:** `select now();`
 - » Output: 2006-12-12 12:21:02
 - **curdate()** - returns current date **Example:** `select curdate();`
 - » Output: 2006-12-12
 - **curtime()** - returns current time **Example:** `select curtime();`
 - » Output: 12:21:24
 - **password()** - returns a hash for the string
Example: `select password("hello");`

Aggregates Functions

- **having** - to deal with aggregate functions (**where** clause cannot be used against aggregates). For example, imaging you have a table where you register the name, and amount spent by a person.

```
create table expenses (name varchar(20), amount int);  
insert into expenses values("Mary", 10);
```

...

- A person can have multiple entries in the table. The following query will provide a list of those that spent more than 40 dollars:

```
select name, sum(amount)  
from expenses  
group by name  
having sum(amount) > 40;
```

limit

- **limit** - controls the number of records returned

Example: `select * from allfriends limit 3;`

- First three records are displayed

Example: `select * from allfriends limit 3,2;`

- Skips the first three records and displays the following two

Joins

- Operation that allow us to combine information from several tables
- **Example:**
 - friends table with fields name, salary, gender
 - foods table with fields person, food
 - If you want to display the name, salary, and food someone likes you can execute the following query:

```
select name, salary, food
from friends, foods
where friends.name = foods.person;
```

- What happens if we remove the where clause?

SQL Root Password

- To change any user's password (including root) execute the following:
use mysql;
update user set password=PASSWORD("NEWPASSWORD") where user='USERNAME';
flush privileges;
- About root accounts
 - There is a root account associated with localhost and one for external accesses
 - Each can have a different password
 - The above update command handles both passwords
- To reset a root password to no password use
 - **update 'root' set password=""**
 - » Notice "" is an empty string
- You can also change passwords through mysqladmin

phpMyAdmin

- Interface to database system
- Just type
 - <http://localhost/phpmyadmin/>
- Alternative
 - <http://www.adminer.org/>